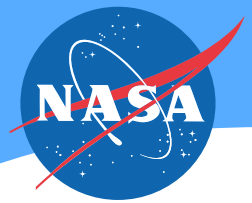
Standardizing the GRACE/GRACE-FO ASCII products with YAML encoding

Wen-Hao Li, Christopher Finch, David Wiese, and Edward Armstrong

2018 GRACE Science Meeting
Potsdam, Germany
9 October, 2018



National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Focuses

1. Motivations for standardizing the GRACE/GRACE-FO ASCII products.
2. Objectives for ASCII File Header Redesign –
   - Improve the metadata
   - Standard ASCII formats.
3. Why adopt the YAML format?
4. YAML File Structure -- Header and Data-Block.
5. Compare YAML header with original.
6. Working with YAML in Python
7. Converting legacy data-block into YAML format.

Motivations for standardizing ASCII files

1. **Meeting NASA requirements**
   a. NASA satellite missions should follow the NASA Earth Science Data Preservation Content Specification, including metadata, and data format.
   b. ESO (ESDIS Standards Office) assists the Earth Science Data and Information System (ESDIS) Project in formulating standards policy for NASA Earth Science Data Systems (ESDS)
   c. ESO has approved ASCII File Format Guidelines for Earth Science Data and required that all ASCII files should be in compliance with this Guidelines.
2. **Standards Improve the GRACE product accessibility and interoperability, making the products FAIR (Findable, Accessible, Interoperable, Reusable)**
   a. Standards increase the data visibility. Standard product names and keywords make the data more Findable
   b. Make the data more accessible, facilitating data modeling and development of service tools.
   c. Standards enable data interoperability, data sharing and maximize the data usage
   d. Increase the data sustainability and reusability.

Objectives for File Header Redesign -- Improve the metadata

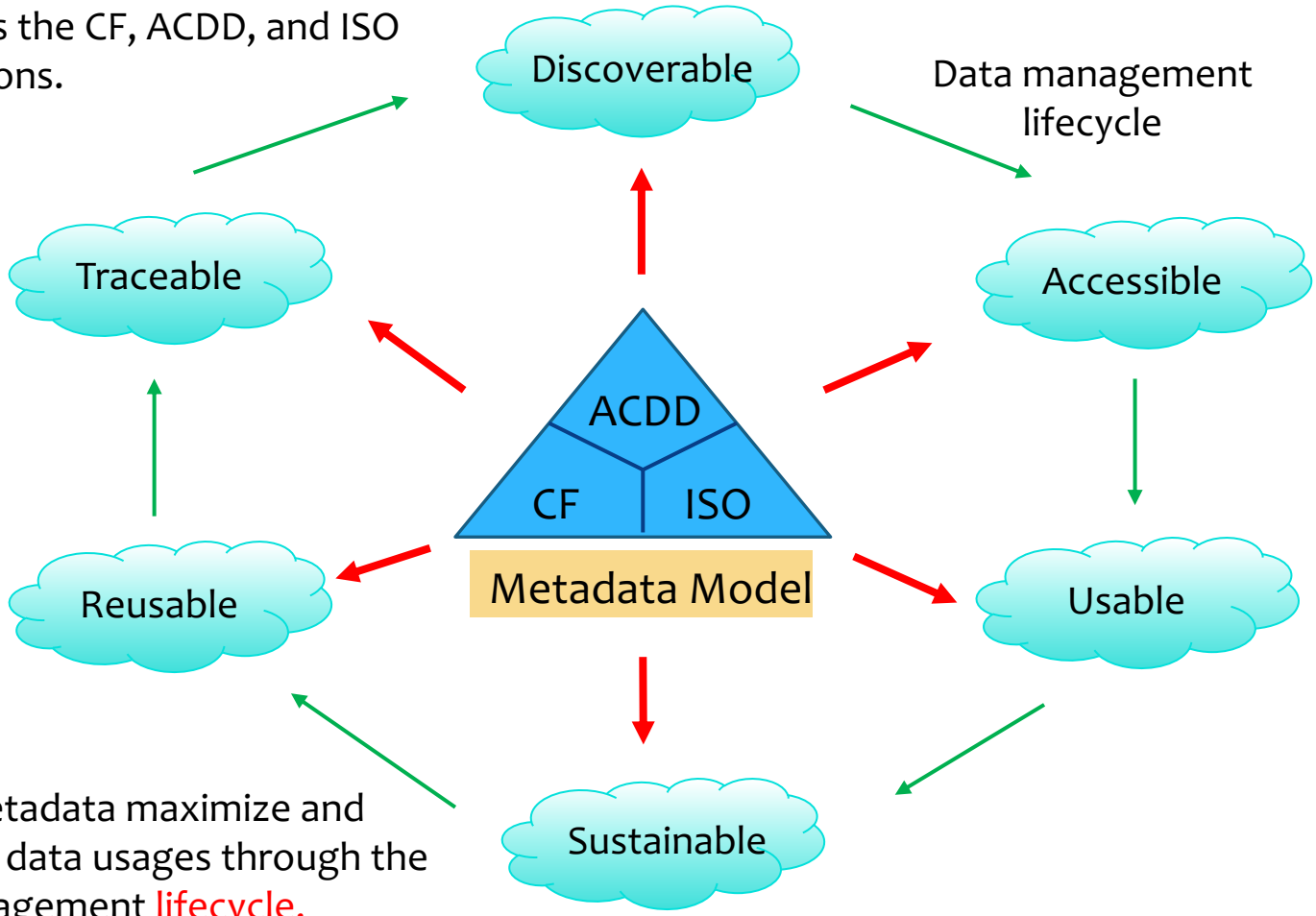
1. **Improve data file metadata**
   - Metadata is the core of the data management lifecycle.
   - Richer metadata improves the data discoverability and accessibility.
   - Richer metadata facilitates the communication between different communities.
   - Metadata is a key measure of data quality.
2. **Increase data interoperability**
   - Standards enable the interoperability.
   - Adopting the NASA-endorsed metadata standards as summarized by PO.DAAC metadata best practices.
   - Climate Forecast (CF), Attribute Conventions for Data Discovery (ACDD) and International Standards Organization (ISO) conventions.

Objectives for File Header Redesign -- Standard ASCII format

3. **Header should be user-friendly, self-explanatory**
   - Easy to read and understand the metadata for all users.
   - Make the data functionally usable and sustainably reusable.
   - Retain legacy information from the original header.
4. **Header structure syntax should be simple**
   - Minimize the effort to implement.
   - Increase header readability.
5. **ASCII file is readable by most common programming languages for data I/O across programs on different platforms.**
   - C/C++, Python, R, Matlab, IDL, Java, Perl, Ruby, PHP, C#.

PO.DAAC Data management vs Metadata Model

- PO.DAAC metadata model combines the CF, ACDD, and ISO conventions.



- Richer Metadata maximize and empower data usages through the data management lifecycle.

Why Adopt the YAML format?

* **YAML encoding is one of the ESDSWG recommended ASCII data formats**
  1. netCDF's NcML (XML representation of netCDF).
  2. netCDF CDL (Common Data Form Language) Text representation of netCDF
  3. JSON (JavaScript Object Notation)
  4. ***YAML*** (YAML Ain't Markup Language)
  5. ICARTT () designed for airborne mission ASCII data.
* **Why choose YAML**
  1. Designed to be human-friendly data serialization language.
  2. Easier to read and write than other common data formats
  3. YAML syntax are simple, clear, and easy to implement
  4. YAML can work with many modern programming languages~~, including C/C++, Python, Java, Perl, PHP, Ruby, C#.~~
  5. Very similar to original header structure.
* **YAML vs JSON**
  1. YAML is closely related with JSON, but simpler.
  2. Technically, YAML is a superset of JSON.
  3. YAML and JSON can be easily converted.

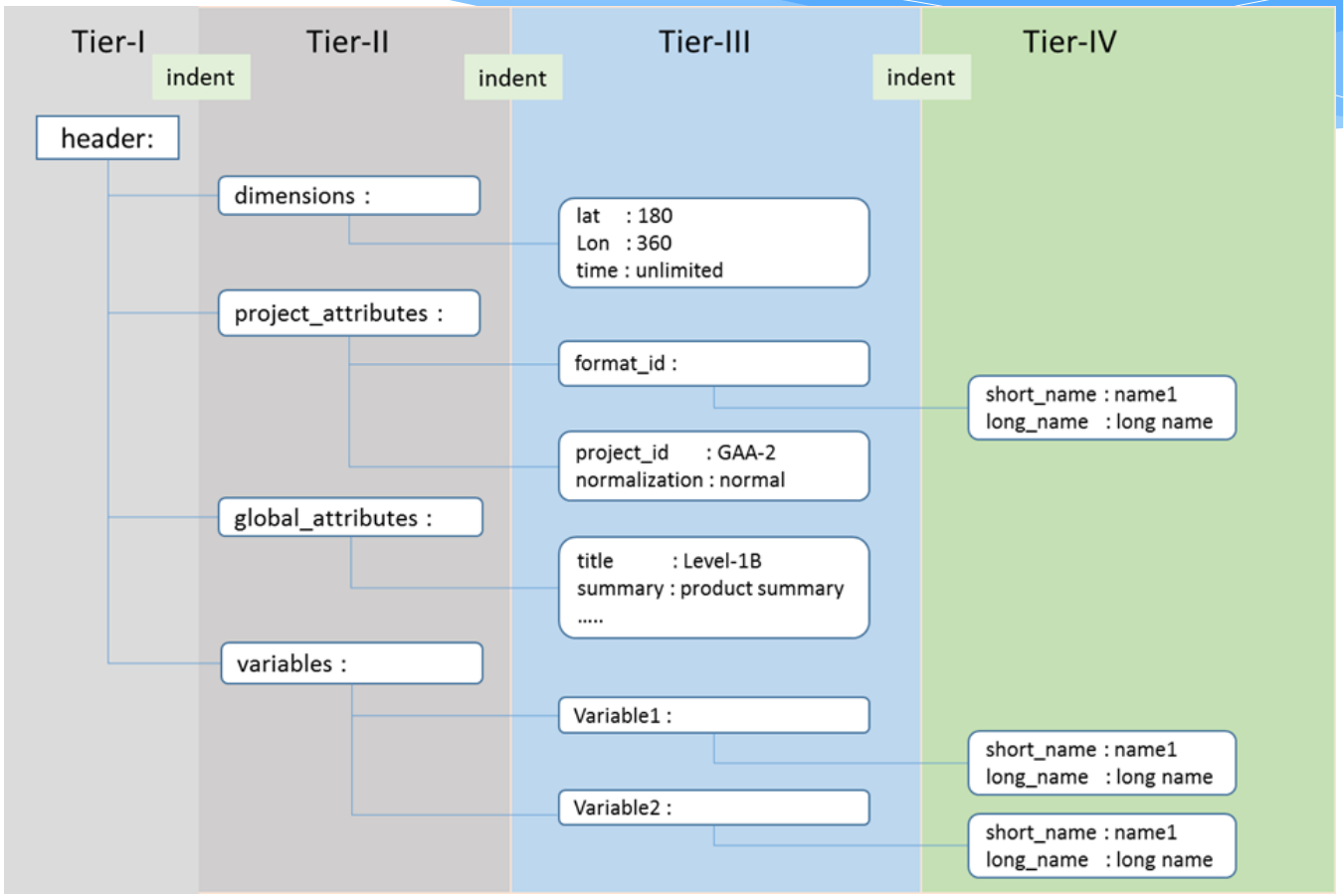
YAML Header Structure

* **Header contains four sections**
  1. **dimensions**
     - Define the spatial and temporal dimensionality, Lat, Lon, and Time
     - In GRACE, it refers to degree and order
  2. **global_attributes**
     - Cover most common standard attributes defined in the NASA standard metadata model
  3. **non-standard_attributes** (project_attributes)
     - Project specific global attributes
  4. **variables**
     - Define data name (standard and long), property for each column.

* **Header contains multiple nested structures (Dictionary in Python)**

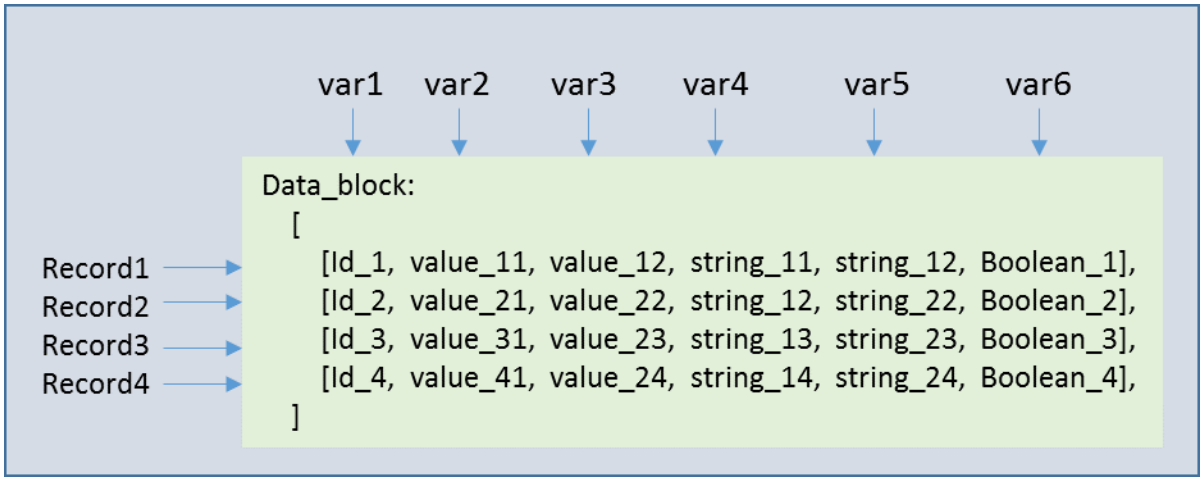
* **YAML header interface is much like CDL (Common Data form Language) format, which is a human-readable text representation of netCDF data.**

Diagram of YAML Header



Standard Data-block design

- Each variable takes a single column
- Each data record takes a single row
- Variable values in each row are separated by comma, and closed with square bracket ([]).
- The data-block should be a 2-D Array/List



Published GRACE Level-2 RL06 products in YAML format

* Recently released the GRACE Level-1/Level-2 RL06 products in YAML format.
  1. GRACE_GAA_L2_GRAV_GRZ_RL06 (JPL,GFZ)
  2. GRACE_GAB_L2_GRAV_GRZ_RL06 (JPL,GFZ)
  3. GRACE_GAC_L2_GRAV_GRZ_RL06 (JPL,GFZ, CDR)
  4. GRACE_GAD_L2_GRAV_GRZ_RL06 (JPL,GFZ, CDR)
  5. GRACE_GSM_L2_GRAV_GRZ_RL06 (JPL,GFZ, CDR)

YAML (rl06) vs original header(rl05)

(GAA-2_2003032-2003059_GRAC-JPLEM_BC01_0600)_

FIRST GAA-2_2003032-2003059_0025_JPLEM_0000_0005 SHM JPL/NASA/USA 20120905
CMMNT Beginning epoch :20030201.0000
CMMNT Center epoch :20030215.0000
CMMNT Ending epoch :20030301.0000
CMMNT The 0th and 1st degree terms are excluded from JPL level-2 processing
CMMNT Unused days (date) : 08-FEB-2003
CMMNT Unused days (date) : 20-FEB-
CMMNT Unused days (date) : 26-FEB-
EART 3.9860044150e+1 6.3781364

Original RL05 Header

dimensions:

Non_standard_attributes:

```
header:
 dimensions:
  degree            : 180
  order             : 180

 non-standard_attributes:
  product_id        : GAA-2
  format_id         :
  short_name        : SHM
  long_name         : Earth Gravity Spherical Harmonic Model Format
  normalization     : fully normalized
  earth_gravity_param   :
   long_name        : gravitational constant times mass of Earth
   units            : m3/s2
   value            : 3.9860044150e+14
  mean_equator_radius   :
   long_name        : mean equator radius
   units            : meters
   value            : 6.3781363000e+06
  comments          : Degree 0 and Degree 1 terms are excluded from JPL Level 2 Processing
```

YAML RL06 Header

global_attributes:

```
global_attributes:
  title             : AOD1B ATM Coefficients JPL RL06
  summary           : |
                      Spherical harmonic coefficients that represent anomalous contributions
                      of the non-tidal atmosphere to the Earth's mean gravity field during
                      the specified timespan.  This includes the contribution of atmospheric
                      surface pressure over the continents, the static contribution of
                      atmospheric pressure to ocean bottom pressure elsewhere, and the contribution
                      of upper-air density anomalies above both the continents and the oceans.
                      The anomalous signals are relative to the mean field from 2003-2014.
  project           : NASA Gravity Recovery And Climate Experiment (GRACE)
  program           : NASA Earth Science System Pathfinder
  keywords          : |
                      GRACE, Level-2, SHM, Spherical Harmonic Model, Gravitational Field, GSM,
                      Geopotential, Gravity Field, Mass, Mass Transport, Total Water Storage,
                      Time Variable Gravity, Mass Balance, Gravity Anomaly, Satellite Geodesy,
                      Atmosphere, Non-Tidal Atmosphere, AOD, Dealiasing Product
  keywords_vocabulary   : NASA Global Change Master Directory (GCMD) Science Keywords
  institution       : NASA/JPL
  conventions       : CF-1.6, ACDD-1.3, ISO 8601
  id                : 10.5067/GRGAA-20J06
  naming_authority  : org.doi.dx
  history           : Original solution produced on 2018-05-19T08:41:19
  source            : All data from AOD1B RL06
  processing_level  : 2
  acknowledgement   : |
                      GRACE is a joint mission of NASA (USA) and DLR (Germany).
                      Use the digital object identifier provided in the id attribute when
                      citing this data.  See https://podaac.jpl.nasa.gov/CitingPODAAC
  license           : https://science.nasa.gov/earth-science/earth-science-data/data-information-policy
  product_version   : 6.0
  references        : ftp://podaac-ftp.jpl.nasa.gov/allData/grace/docs/AOD1B_PDD_RL06_v6.0.pdf
  creator_name      : GRACE Science Data System NASA/JPL
  creator_email     : grace@podaac.jpl.nasa.gov
  creator_url       : https://grace.jpl.nasa.gov
  creator_type      : group
  creator_institution   : NASA/JPL
  publisher_name    : Physical Oceanography Distributed Active Archive Center
  publisher_email   : podaac@jpl.nasa.gov
  publisher_url     : https://podaac.jpl.nasa.gov
  publisher_type    : group
  publisher_institution : NASA/JPL
  time_coverage_start   : 2003-02-01T00:00:00.00
  time_coverage_end     : 2003-02-28T23:59:59.00
  unused_days       : [ 2003-02-08, 2003-02-20, 2003-02-21, 2003-02-26 ]
  date_created      : 2018-05-19T08:41:19
  date_issued       : 2018-05-19T08:41:19
```

variables:

```
# EGM Coefficient Records 2
variables:
  record_key:
    key_name        : GRCOF2
    long_name       : Earth Gravity Spherical Harmonic Model Format Type
    coverage_content_type : referenceInformation
    data_type       : string
    comment         : 1st column
  degree_index:
    long_name       : spherical harmonic degree l
    coverage_content_type : referenceInformation
    data_type       : int32
    comment         : 2nd column
  order_index:
    long_name       : spherical harmonic order m
    coverage_content_type : referenceInformation
    data_type       : int32
    comment         : 3rd column
  clm:
    long_name       : Clm coefficient; cosine coefficient for degree l and order m
    coverage_content_type : modelResult
    data_type       : double precision
    comment         : 4th column
  slm:
    long_name       : Slm coefficient; sine coefficient for degree l and order m
    coverage_content_type : modelResult
    data_type       : double precision
    comment         : 5th column
  clm_std_dev:
    long_name       : standard deviation of Clm
    coverage_content_type : qualityInformation
    data_type       : double precision
    comment         : 6th column
  slm_std_dev:
    long_name       : standard deviation of Slm
    coverage_content_type : qualityInformation
    data_type       : double precision
    comment         : 7th column
  epoch_begin_time:
    long_name       : epoch begin of Clm, Slm coefficients
    time_format     : yyyymmdd.hhmm
    coverage_content_type : referenceInformation
    data_type       : string
    comment         : 8th column
  epoch_stop_time:
    long_name       : epoch stop of Clm, Slm coefficients
    time_format     : yyyymmdd.hhmm
    coverage_content_type : referenceInformation
    data_type       : string
    comment         : 9th column
  solution_flags:
    long_name       : Coefficient adjustment and a priori flags
    coverage_content_type : auxiliaryInformation
    data_type       : byte
    flag_meanings   :
     - char 1 = Clm adjusted, y for yes and n for no
     - char 2 = Slm adjusted, y for yes and n for no
     - char 3 = stochastic a priori info for Clm, y for yes and n for no
     - char 4 = stochastic a priori info for Slm, y for yes and n for no
    comment         : 10th column
  solution_comment:
    long_name       : Comment when present
    coverage_content_type : referenceInformation
    data_type       : string
    comment         : 11th column
# End of YAML header
```

Working with YAML in Python

* **Open/read YAML file**

```
1 > import yaml
2 > stm = open("full_path/filename.yaml", 'r')
3 > data = yaml.load(stm)
```

* **Output YAML file**

```
1 > out = open("full_path/filename.yaml", 'w')
2 > yaml.dump(data, out, indent = 4, default_flow_style=False)
```

* **Convert YAML to JSON**
  * YAML is superset of JSON

```
1 > import sys
2 > import json
3 > with open('outfile.json', 'w') as out1:
        json.dump(data,out1, indent=4, separators=(',', ': '))
```

Working with YAML in Python

* **Extract the metadata information**
  * Following the nested structure down to the element

```
1   > undays = data['header']['global_attributes']['unused_days'][0]
2   > print ('Unused_days =', undays)
3   > Unused_days = 2003-02-08
```

* **Extract the data record values**
  * Depend on the data-block structure
  * Standard data-block

```
1   > data['datablock2'][0][3]      # value at 1st row and 4th col
2   > -1.00683e-11                  # result
```

Convert Legacy GRACE data-block to YAML format compatible.

- GRACE legacy data-block remain unchanged in current YAML ASCII files.
- Converted to YAML format by adding three lines of simple codes.

3.2.2 GRACE legacy data-block:

```
GRCOF2   0   0 -1.0068e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   1   0 -2.6640e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   1   1  3.5867e-11  7.1248e-11  0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   0  6.0144e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   1  7.6731e-12  3.949e-11    0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   2 -3.702e-12    1.8947e-11  0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
```

3.2.3 Convert to YAML formatted data-block:

```
Data-block :
 [
GRCOF2   0   0 -1.0068e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   1   0 -2.6640e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   1   1  3.5867e-11  7.1248e-11  0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   0  6.0144e-11  0.0000e+00 0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   1  7.6731e-12  3.949e-11    0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
GRCOF2   2   2 -3.702e-12    1.8947e-11  0.0000e+00 0.0000e+00 20030201.0000 20030301.0000 nnnn
 ]
```

Extract the data value

* Extract the value from the data-block at any given location
* Parsing is a little complicated

```
1   > coef = data['grace_data']      # assign the grace data to coef
2   > nList = coef[0].split()        # split the single string with space
3   > print(nList[3])                # print value at 1st row and 4th col
4   > -1.00683e-11                   # result
```

Thank You